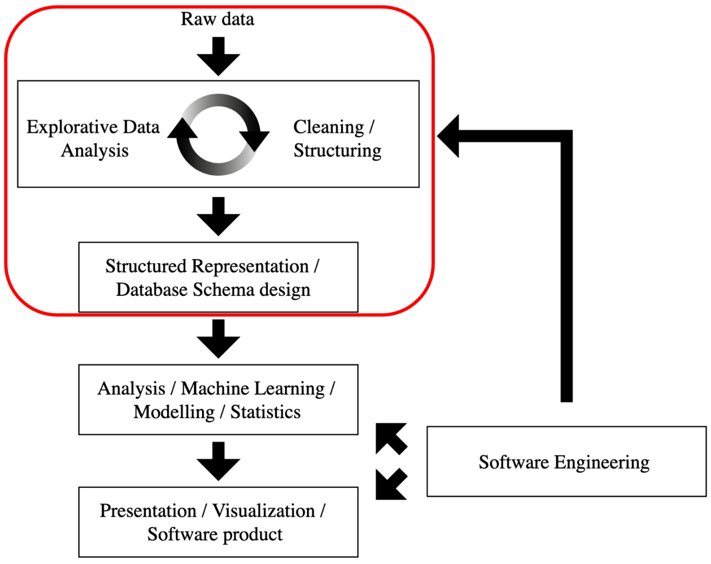
Individual Pass/Fail Exercise 2

Start Assignment

- Due Monday by 16:00
- Points 0
- Submitting a file upload
- File types ipynb and pdf
- Available 24 Feb at 8:00 - 10 Mar at 17:00

Web Scraping for Fake News Detection

In this assignment, you'll gain hands-on experience in web scraping, a crucial skill in data science, especially when structured datasets are not readily available. Specifically, you'll focus on extracting information from news websites, a vital step in creating a dataset for training a fake news detection model. In terms of the Data Science Pipeline, you will mainly focusing on acquiring raw data, processing data, cleaning and explorative data analysis, and structured representation and storage of data.


Raw data
Explorative Data Analysis
Cleaning / Structuring
Structured Representation / Database Schema design
Analysis / Machine Learning / Modelling / Statistics
Presentation / Visualization / Software product
Software Engineering

Revision History

Thursday 27 February	Revised the assignment description to instruct students to now use Selenium to fetch 800 articles. Submission deadline pushed back by three calendar days to Monday 10th March.

Submission Requirements

Jupyter Notebook (.ipynb file) implementing the assignment.

PDF printout of the executed Jupyter Notebook displaying the results.

Part 1: Analyze the Fake News Dataset

1. Import Dataset: Import the cleaned dataset from last assignment
2. Dataset Analysis:

- Determine which article types should be omitted, if any.
- Group the remaining types into 'fake' and 'reliable'. Argue for your choice.
- Examine the percentage distribution of 'reliable' vs. 'fake' articles. Is the dataset balanced? Discuss the importance of a balanced distribution.

Additional Resources: More information about the dataset is available on **Github (https://github.com/several27/FakeNewsCorpus)** .

Part 2: Gathering Links

In this part of the exercise you will write code to extract a collection of article links.

1. Library Installation: Install selenium (`pip install selenium`). Create a new Jupyter Notebook and import this module: `from selenium import webdriver`
2. Retrieve HTML Content: Use the following example code to fetch the HTML content of a webpage and verify that contents holds the HTML source of the webpage:

```
browser = webdriver.Firefox()
browser.get('https://www.bbc.com/news/world/europe')
```

3. Extract Articles: Selenium allows us to extract information easily. You can read the documentation here: **https://selenium-python.readthedocs.io/index.html (https://selenium-python.readthedocs.io/index.html)** **(https://www.selenium.dev/documentation/)** . Write a function to extract all articles from the page using the `find_elements` method. For each article, retrieve the headline, the summary, and the link to the article (`href`). (Hint: You might have to inspect the page for `class`es, `id`s, etc. that might be useful for extracting the information)
4. Scrape Multiple Pages: Identify the number of pages available for the 'Europe' section (Hint: See buttons at the bottom of the page). Write a function that extracts all article links from all these pages (Hint: Click on button using `.click().` You may also have to add a delay for the page to load using `time.wait(n_seconds))`
5. Expand the Scope: Extend your scraping to include articles from other regions: US & Canada, UK, Australia, Asia, Africa, Latin America, and the Middle East. If done correctly, you should get around 800 article links.
6. Save Your Results: Store the collected links in a file (CSV, JSON, or TXT format).

Part 3: Scraping Article Text

In this final part of the exercise, you will scrape the article text and store it on disk.

1. Article Inspection: Manually inspect a few articles to find unique attributes to identify the text, the headline, the published date, and the author.
2. Text Scraping Function: Implement a function that takes a URL and returns a dictionary with the article's text, headline, published date, and author.

3. Scrape All Articles: Loop through all the collected article links to scrape their contents (May take a long time. So try on a smaller subset to start with). Remember to implement error handling and possibly introduce delays to avoid being blocked.
4. Data Storage: Save the scraped article data to a file.
5. Discussion: Discuss whether it would make sense to include this newly acquired data in the dataset. Argue why or why not and if possible include statistics to support your claim.

Part 4: Preservation

Keep the data that you have scraped so you can use it for your Group Project!